REMARKS

By this amendment, Claims 10, 21, 32, and 43 are amended and no claims are added or canceled. Hence, Claims 10, 12, 14, 16-21, 23, 25, 27-32, 34, 36, 38-44, 46, and 48-56 are pending in the application. The amendments to the claims as indicated herein do not add any new matter to this application.

Each issue raised in the Office Action mailed August 29, 2008 is addressed hereinafter.

I. PRIOR ART ISSUES

Claims 10, 12, 14, 16-21, 23, 25, 27-32, 34, 36, 38-44, 46, 48-56 stand rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over U.S. Patent No. 5,008,814 issued to Mather ("*Mathur*"), in view of alleged Admitted Prior Art ("APA") and U.S. Patent Application No. 2005/0198629 to Vishwanath ("*Vishwanath*"). This rejection is respectfully traversed.

A. CLAIM 10

Claim 10 now recites:

> A method of software loading and initialization in a distributed network of nodes, the method comprising:
> persistently storing, in a first storage of a master node, a plurality of software packages and a plurality of boot images, wherein the plurality of software packages and the plurality of boot images will be used by the nodes in the distributed network;
> persistently storing, in a second storage of the master node, software version information and node type information for each node in the distributed network;
> receiving, at the master node, a request for a boot image and software packages from a node, in the distributed network, that is performing an initial boot;
> based on the request, the master node determining software version information of the node to retrieve from the second storage;
> the master node retrieving the software version information of the node from the second storage;
> the master node determining, based on the software version information of the node, a boot image of the plurality of boot images and one or more software packages of the plurality of software packages to extract from the first storage;

the master node extracting the boot image and the one or more software packages from the first storage;
the master node delivering, to the node, the boot image and the one or more software packages;
wherein said node:
(a) stores the boot image and the one or more software packages in its local persistent storage,
(b) extracts software version information from the one or more software packages and stores the software version information in the local persistent storage,
(c) reboots and executes the boot image stored in the local persistent storage, and
(d) in response to executing the boot image, verifies the software version information with said master node by sending the software version information to the master node;
the master node receiving the software version information from the node;
in response to receiving the software version information, the master node determining whether the node has the correct software versions;
in response to the master node determining that said node does not have the correct software versions, then the master node retrieving correct software packages from the first storage and sending the correct software packages to said node, wherein said node stores the correct software packages in the local persistent storage and completes booting by executing the correct software packages stored in the local persistent storage. (emphasis added)

At the least the above-bolded features of Claim 10 are not taught or suggested by the cited art.

1. The cited art fails to teach or suggest that a node that receives software packages verifies software version information with the master node

Claim 10 now recites that "said node…(d) in response to executing the boot image, verifies the software version information with said master node by sending the software version information to the master node" (emphasis to show added claim language). On page 4, the Office Action cites col. 8, lines 34-56 of *Mathur* for allegedly disclosing "wherein said node…verifies the software version information with said master node." This is incorrect. Col. 8, lines 23-56 state:

> In the preferred embodiment, the softload process is implemented by a master task and a slave task running in the CPU's 101 of each node. When a "distribution command" is entered at the operator node N_O, the master task at that node will accept the command, and based upon the context thereof, relays appropriate

> messages to the slave task of the source node N_S and the respective slave tasks of the destination nodes N_d.
>
> At the source node N_S, the master task translates the messages into directives, and send them to its slave task. Responsive to the directives, the slave task obtains information about the non-volatile storage devices 103 and **sends this information back to the operator** node N_O for display thereat.
>
> The slave tasks at the destination nodes N_d also collect information about their non-volatile storage devices 103 (e.g. **software versions**, statuses, or whether a non-volatile storage device 103 can be programmed) and send this information back to the operator node N_O.
>
> The information from the source node N_S and destination nodes N_d are **used by the master task** at the operator node N_O to provide the initial checking and selection to exclude nodes that are not appropriate to be a source node for the softload process (e.g. nodes which are currently performing a softload process, nodes currently acting as source nodes for other softload processes, or nodes that do not contain a "new" non-volatile storage device 103, etc.). The master task at the operator node N_O may also exclude nodes which are not appropriate to be destination nodes N_d (e.g. nodes with no communication links connected to the source node, nodes that are not operational, or nodes which have no non-volatile storage device 103, etc.). (emphasis added)

The Office Action quoted the above-bolded portion of *Mathur* for allegedly disclosing that the node verifies software version information with the master node. There are multiple reasons why this is incorrect. First, according to Claim 10, the software version information that a node sends to the master node is concerning software that the node receives from the master node. In contrast, according to *Mathur*, the software version information that an operator node receives from a destination node is done in preparation for a new software distribution.

Second, the verification of the software version information between the node and master node (according to Claim 10) occurs in response to the node executing a boot image that the node received from the master node. In contrast, *Mathur* states that the destination node sends the software version information in response to a request from the operator node.

Third, according to Claim 10, the master node uses the software version information to determine whether the node has the correct software versions. In contrast, *Mathur* is silent with respect to what the operator node (i.e., the alleged master node) even does with the software

version information received from the destination node. Thus, *Mathur* fails to teach or suggest that the software version information is even verified.

2. *The cited art fails to teach or suggest what happens if the software versions of software packages that a node receives are not correct*

The Office Action concedes (on page 6) that *Mathur* fails to disclose:

> if said node does not have the correct software versions, then the master node retrieving correct software packages from the first storage and sending the correct software packages to said node, wherein said node stores the correct software packages in the local persistent storage and completes booting by executing the correct software packages stored in the local persistent storage

as previously recited in Claim 10. The Office Action then asserts, "Mathur discloses verification if correct version is stored prior to distribution then consistency is checked after it has been distributed." The Office Action cites col. 5, line 31- col. 6, line 10 of *Mathur* as evidence of this assertion. This is incorrect. The pertinent portion of *Mathur*, of which the Office Action quotes a part, states:

> When the source node N_S receives a distribution message, **a check is made** to see: (1) **whether** the source node N_S is currently being installed with a **new system software version**, (2) whether the source node N_S has been designated as the source node N_S of another softload process, and (3) whether the source node N_S has a "new" non-volatile storage device 103 at the specified location. If (1) and (2) are false and (3) is true, a "consistency check" will be performed on the "new" non-volatile storage device 103 (block 202). In the preferred embodiment, the consistency check involves a process of making a checksum verification on the "new" software, as well as checking the actual number of modules of the system software contained in the non-volatile storage device 103. This number of modules is compared against the number stored in the header of the non-volatile storage device 103. (emphasis added)

The Office Action quoted only the above-bolded portion of *Mathur*. According to *Mathur*, the source node (i.e., not the destination/target node) checks itself (i.e., not another node as Claim 10 requires) to determine whether a new software version is currently being installed. If so, then the source node does not take part of in the software distribution. In contrast, according to Claim 10, the determination of whether a node has correct software packages comes after the software

packages have been delivered to the node. Further, it is unsurprising that because this cited portion fails to teach or suggest anything about software version information, *Mathur* does not disclose "verification if correct version is stored prior to distribution."

The Office Action asserts that:

> Mathur discloses…maintaining dialogue between a node trying to [IPL] and a operator node using a master node intermediary (Fig. 5-6; col. 7 line 40 to col. 8 line 22; col. 8 lines 34-56) **whereby to follow progress of update, as to for example validating on correctness of version** being tried at the target node can be made via acknowledgement received from target node, wherein this continual acknowledging scheme would also assure retransmission of data based on acknowledgement of proper reception at the receiving node (col. 9 lines 12-20). (emphasis added)

The Office Action's allegation that *Mathur* discloses "validating on correctness of version" suggests that the target node or operator node validates software that has been distributed using software version information. This is incorrect. *Mathur* does not use software version information for validating the correctness of a version. Instead, *Mathur* provides that (a) a target node performs a consistency check (on itself) that comprises comparing checksums and the number of modules, (b) if the consistency check is successful, the node performs an IPL, (c) if the IPL is successful, then the a dialogue is initiated between the target node and the operator node (see col. 7, line 31- to col. 8, line 23). Also, col. 9, lines 12-20 of *Mathur* states that during distribution of software, if a target node detects that a packet has not been delivered, then the target node sends a retransmission request to the source node. In all, these cited portions of *Mathur* fail to teach or suggest using software version information in any way.

Moreover, the underlined statement above, i.e., "wherein this continual acknowledging scheme would also assure retransmission of data based on acknowledgement of proper reception at the receiving node," is incorrect. Data would not be retransmitted to a receiving/target node if the receiving node acknowledged proper reception.

On page 7, the Office Action states, "(see Fig 17 [of *Vishwanath*] – Note: incremental provisioning **reads on** retrieving correct versions if acknowledge that previous sent version is not satisfactory)." (emphasis in Office Action). Such an interpretation of "incremental provisioning" is overly broad and unreasonable. *Vishwanath* lacks any notion that a target node indicates (to a source or master node) that a version that the target node received is unsatisfactory. Therefore, it follows that *Vishwanath* cannot teach or suggest retrieving correct versions if a target node acknowledges that the previous sent version is not satisfactory.

The Office Action further asserts:

> Based on upgrade of versions as set forth in Mathur and APA, such that proper components supporting completion of a node booting into a working state at a target node, and **Mathur's maintained checking using the master node and operator regarding correctness of downloaded version in order for it to be permanently stored** for activation at the target node (steps 202 – 204 Fig. 2), it would have been obvious for one skill in the art at the time the invention was made to implement Mathur's communication between the target node and the Master node (using operator directives) so that acknowledging of proper versioned package be implemented until all proper versionsed upgrade components have been incrementally provisioned or retransmitted using Vishwanath's approach as to fulfill the provision of upgrade components until all the required components for Mathur's node to ILP into a proper O.S. state; that is, **when the node is not satisfied with the received components (as taught in Vishwanath) – para 0155, pg. 16), the additional ILP components or more correct version being retrieved by the server** or by invoking other provisioning services, so that **more correct components be retransmitted to the target node**, in order for Mathur's target node to complete the ILP intended via the above dialogue and acknowledgement approach, whereby a trial version can be finalized as complete and successful, based on the incremental provisioning by Vishwanath and the maintained update status communication approach by Mathur. (emphasis added)

This is incorrect. Steps 202 – 204 in FIG. 2 of *Mathur* do not disclose that the source and operator nodes in *Mathur* maintain checking regarding the correctness of downloaded software. The software distribution does not even occur until step 204. In step 205, a target node will send a message to the source node if the target node detects that it is missing a packet. The source node does not affirmatively check the correctness of the downloaded software. Following

distribution, the operator node issues a cutover command (step 207) to the target node, which does not include checking the correctness of downloaded software. At step 208, the target node performs a cutback if it fails to IPL (i.e., reboot) correctly. If the target node IPLs correctly, then, at step 209, the target node performs a trial use of the downloaded software, which includes one or more exchanges of acknowledgements between the target node and the operator node.

If the Office Action is alleging that this exchange of acknowledgement constitutes maintaining checking regarding the correctness of downloaded software, the Office Action appears to overlook other claimed features. According to Claim 10, the destination node does not complete rebooting and execute the correct software packages until after the node verifies the software version information with the master node. In contrast, in *Mathur* a trial use of the new system is performed before the alleged maintained checking of correctness of downloaded software is performed.

The Office Action also incorrectly alleges that "when the node is not satisfied with the received components (as taught in Vishwanath) – para 0155, pg. 16), the additional ILP components or more correct version being retrieved by the server." Paragraph 155 of *Vishwanath* merely states that if a target server does not satisfy one or more constraints associated with the group of target servers, then the system may be paused and input from an administrator may be requested. An example of a constraint (i.e., Constraint 1) is found in paragraph 0109 of *Vishwanath*: "RAM>=512 MB, Hard Disk Size>=20 GB, Processor>=Pentium 3, Processor Speed>=1000 Hz." Therefore, in order for a target server to take part in a software distribution process, the target server (depending on the specific policy) may have to satisfy this or similar hardware constraints. If a target server does not satisfy a constraint, then the target server is not included in the software distribution process. Therefore, *Vishwanath* does not teach or suggest that "additional ILP components or more correct version"

are retrieved by the server if the target server does not satisfy a constraint, as alleged in the Office Action.

The Office Action incorrectly alleges that "when the node is not satisfied with the received components…the additional ILP components or more correct version being retrieved by the server…so that more correct components be retransmitted to the target node" (emphasis added). At best, *Mathur* provides that if a particular packet has not been received at a target node, then the target node requests the source node to retransmit that packet. In *Mathur*, "more correct components" may indicate different components than were previously transmitted. However, both *Mathur* and *Vishwanath* fail to teach or suggest that different components are transmitted to a target node if the target node is not satisfied with previously received components. Furthermore, such an interpretation is inconsistent with the "be retransmitted" language that the Office Action employs, because different components that have not yet been transmitted cannot be ***re***transmitted. In *Mathur*, "more correct components" may indicate additional components that were previously transmitted. However, if a target node detects that it failed to receive a particular packet, then the target node requests that packet from the source node.

Therefore, the Office Action includes multiple misstatements that incorrectly interpret the cited references. In other words, the attempts to correlate features of the cited references with features of Claim 10 rely on an incorrect interpretation of *Mathur* and *Vishwanath*.

Fundamentally, *Mathur*, APA, and *Vishwanath* fail to teach or suggest that: (1) a node verifies software version information, of a software package that the node receives, with a master node; (2) the master node uses the software version information to determine whether the node has the correct software versions; and (3) in response to determining that the node does not have the correct software versions, the master node retrieves the correct software packages and sends

them to the node. Because neither *Mathur*, APA, nor *Vishwanath* teaches or suggests any of these features of Claim 10, the combination of *Mathur*, APA, and *Vishwanath* also fails to teach or suggest these features of Claim 10.

Based on the foregoing, Claim 10 is patentable over the cited art. Reconsideration and withdrawal of the rejection of Claim 10 under 35 U.S.C. § 103(a) is therefore respectfully requested.

B. CLAIMS 21, 32, AND 43

Each of independent Claims 21, 32, and 43 is either a computer-readable storage medium, an apparatus, or a system claim. Each of Claims 21, 32, and 43 recites the features discussed above that distinguish present Claim 10 over the cited art. Therefore, each of Claims 21, 32, and 43 is patentable over the cited art for at least the reasons given above for Claim 10.

C. CLAIMS 16, 27, 38, AND 48

Each of Claims 16, 27, 38, 48 recites that the master node has the ability to categorize nodes into classes where all of the nodes in a particular class of nodes have the same software configuration. The Office Action (again) cites col. 5, lines 3-27 and col. 3, lines 29-53 of *Mathur* for allegedly disclosing this feature. The Office Action further states, "Note: structure representing grouping of nodes according to some particular privilege configuration reads on class of nodes of same configuration but different processor." This is incorrect.

The cited portion of *Mathur* states:

> Advantageously, the system is implemented so that a distribution command is recognizable by a controller 100 only when it is issued with predefined **access privileges**. Furthermore, **such privileges** may be defined with an hierarchical structure so that the privilege required for a distribution process depends upon the importance of the software.
>
> (col. 5, lines 21-27; emphasis added)

Thus, the privileges referred to in *Mathur* and in the Office Action are **access** privileges that are used to determine whether a distribution command (which identifies a source node and one or more destination nodes) will be recognizable, and thus executable. Contrary to the assertion in the Office Action, *Mathur* does **not** suggest a "structure representing grouping **of nodes**" (emphasis added). The Office Action fails to cite any portion of *Mathur* for teaching or suggesting such a structure or grouping **of nodes**. In fact, the cited portion of *Mathur* states the prior art approach of requiring a human user to identify each destination node of a software update (see col. 5, lines 1-3, 6-7, and 12-14). *Vishwanath* is not cited for teaching or suggesting this feature of Claims 16, 27, 38, and 48.

In the "Response to Arguments" section, the Office Action asserts that "Applicant's arguments fail to comply with 37 C.F.R. § 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the reference." Applicants's arguments have specifically pointed out how the language of the claims patentably distinguishes the claim from *Mathur*. For example, as discussed above, Applicants clearly show how *Mathur* does not even suggest a **group of nodes**, as the Office Action alleges.

Also in the "Response to Arguments" section, the Office Action states:

> The Office Action has interpreted **grouping in classes or categories in the way that some privileges can be grouped together** in order to allow only a same privilege level or group of nodes (i.e., commonly having such level) to be disseminated with the upgrades by *Mathur*. It is the burden of the Applicants to provide convincing facts as to show how the interpretation would be inappropriate, and how the language of the limitation has to be construed otherwise. (emphasis added)

Thus, the Office Action is alleging that it is reasonable to equate the defining of access privileges in a hierarchical structure (as in *Mathur*) with a master node's ability to categorize **nodes** into classes where all of the nodes in a particular class have the same software configuration. This is

incorrect. Access privileges that are defined in a hierarchical structure are completely unrelated to a master node's ability to categorize nodes into classes, which nodes will be the recipients of a software distribution. The hierarchical structure of access privileges has nothing to do with the target nodes. Instead, the hierarchical structure of access privileges is merely used to determine whether software will be distributed at all. Also, there is nothing in *Mathur* to even suggest the fact that nodes of a particular class have the same software configuration. Thus, the Office Action reads subject matter into *Mathur* that simply does not exist and does not follow from a reasonable reading of the text. Consequently, the Office Action's interpretation of Claim 10 and *Mathur* is unreasonable.

Based on the foregoing, the cited art fails to teach or suggest all the features of Claims 16, 27, 38, and 48. Therefore, each of Claims 16, 27, 38, and 48 is patentable over the cited art. Reconsideration and withdrawal of the rejection of Claims 16, 27, 38, and 48 under 35 U.S.C. § 103(a) is therefore respectfully requested.

D. REMAINING DEPENDENT CLAIMS

The dependent claims not discussed thus far are dependent claims, each of which depends (directly or indirectly) on one of the independent claims discussed above. Each of the dependent claims is therefore patentable over the cited art for at least the reasons given above for the claim on which it depends. In addition, each of the dependent claims introduces one or more additional limitations that may independently render it patentable. However, due to the fundamental differences already identified and to expedite the positive resolution of this case, a separate discussion of those limitations is not included at this time. The Applicants reserve the right to further point out the differences between the cited art and the novel features recited in the dependent claims.

II. CONCLUSIONS & MISCELLANEOUS

For the reasons set forth above, all of the pending claims are now in condition for allowance. The Examiner is respectfully requested to contact the undersigned by telephone relating to any issue that would advance examination of the present application.

A petition for extension of time, to the extent necessary to make this reply timely filed, is hereby made. If applicable, please charge the deposit account for the petition for extension of time fee. If any applicable fee is missing or insufficient, throughout the pendency of this application, the Commissioner is hereby authorized to change any applicable fees and to credit any overpayments to our Deposit Account No. 50-1302.

Respectfully submitted,
HICKMAN PALERMO TRUONG & BECKER LLP

Dated: October 7, 2008

/DanielDLedesma#57181/
Daniel D. Ledesma
Reg. No. 57,181

2055 Gateway Place Suite 550
San Jose, California 95110-1083
Telephone No.: (408) 414-1080 ext. 229
Facsimile No.: (408) 414-1076